Homework 3

计 52 宋世虹 2015011267

2017 年 6 月

1 对象与函数

对象：

- Node: 表示 seam dp 时每一个点的权值和路径中上一个点的值。
- type：表示当前要 carve 的是列还是行。
- newPoint：表示 cut 完的 seam 中的一个点。
- T：双向缩放的时候所用的数据结构，其中存有：当前图片，seam 掉一行或一列后的图片，seam 掉一行的路径，seam 掉一列的路径，seam 掉一行的 cost，seam 掉一列的 cost，上一次 choose 了 seam 掉一行还是一列，当前的图片的 seam 权值。

函数：

- inImage：判断一个点是否在图片内。
- calculateEnergy：计算论文中所述能量，可以用不同算子。
- DP：通过 energy 函数计算 seam，dp 出全图每点的权值，存在 seam 中。
- calculateMin：根据 seam 图计算最小的 seam，返回最小的那一列（行）标。
- removeline：根据已得的 seam 移去一行或一列。
- addLine：根据已得的 seam 重新在删去 seam 的图中加上一条红色的 seam。
- getInfo：对于一个已知图片的 T，计算它的行列 seam。
- onMouse：将鼠标选中的区域染红。

2 seam 的求解

使用上述 calculate energy 和 dp 函数即可。

代码：

```
void calculateEnergy(Mat& image,double** energy,int row,int col)
{
  Mat imageXY8UC = image.clone();
  Mat imageX=Mat::zeros(image.size(),CV_8UC3);
    Mat imageY=Mat::zeros(image.size(),CV_8UC3);
    Mat imageXY=Mat::zeros(image.size(),CV_8UC3);
    Mat imageX8UC;
    Mat imageY8UC;
//    for(int i=0;i<image.rows;i++)
//    {
//        for(int j=0;j<image.cols;j++)
```

```
//         {
//             //通过指针遍历图像上每一个像素
//             for (int k = 0;k < 3;++k)
//               {
//                imageX.at<Vec3b>(i,j)[k] =jueduizhi(
//           (i > 0 && j < image.cols-1 ? image.at<Vec3b>(i-1,j+1)[k] : 0)
//           + (j < image.cols-1 ? image.at<Vec3b>(i,j+1)[k]*2 : 0)
//           + (i < image.rows-1 && j < image.cols-1 ? image.at<Vec3b>(i+1,j+1)[k] : 0)
//           - (i > 0 && j > 0 ? image.at<Vec3b>(i-1,j-1)[k] : 0)
//           - (j > 0 ? image.at<Vec3b>(i,j-1)[k]*2 : 0)
//           - (i < image.rows-1 && j > 0 ? image.at<Vec3b>(i+1,j-1)[k] : 0)
//         );
//                imageY.at<Vec3b>(i,j)[k]=jueduizhi(
//           (i < image.rows-1 && j > 0 ? image.at<Vec3b>(i+1,j-1)[k] : 0)
//           + (i < image.rows-1 ? image.at<Vec3b>(i+1,j)[k]*2 : 0)
//           + (i < image.rows-1 && j < image.cols-1 ? image.at<Vec3b>(i+1,j+1)[k] : 0)
//           - (i > 0 && j > 0 ? image.at<Vec3b>(i-1,j-1)[k] : 0)
//           - (i > 0 ? image.at<Vec3b>(i-1,j)[k]*2 : 0)
//           - (i > 0 && j < image.cols-1 ? image.at<Vec3b>(i-1,j+1)[k] : 0)
//         );
//           }
//         }
//     }
//     addWeighted(imageX,0.5,imageY,0.5,0,imageXY);//融合X、Y方向
//     convertScaleAbs(imageX,imageX8UC);
//     convertScaleAbs(imageY,imageY8UC);
//     convertScaleAbs(imageXY,imageXY8UC);    //转换为8bit图像
          for(int i=0;i<image.rows;i++)
        {
            for(int j=0;j<image.cols;j++)
            {
                //通过指针遍历图像上每一个像素
                for (int k = 0;k < 3;++k)
                  {
                    imageXY8UC.at<Vec3b>(i,j)[k] =jueduizhi(
                - (i > 0 ? image.at<Vec3b>(i-1,j)[k] : 0)
                - (i < image.rows-1 ? image.at<Vec3b>(i+1,j)[k]*2 : 0)
                - (j < image.cols-1 ? image.at<Vec3b>(i,j+1)[k] : 0)
                - (j > 0 ? image.at<Vec3b>(i,j-1)[k] : 0)
                + (image.at<Vec3b>(i,j)[k]*4)
              );
                  }
            }
          }
    for (int i = 0;i < row;++i)
    {
      for (int j = 0;j < col;++j)
      {
        energy[i][j] = 0;
        for (int k = 0;k < 3;++k)
        {
          energy[i][j] += (int)(imageXY8UC.at<cv::Vec3b>(i,j)[k]);
        }
        // energy[i][j] += (int)(imageXY8UC.at<uchar>(i,j));
      }
    }
}
```

calculatEnergy 用了不同的算子，会在后文中提到，energy 会用这三通道的输出结果的直接和作为真正的 energy。

DP 计算了论文中所给的方法，每次选择上面的左中右三个 pixel 的 seam 权值，来得到当前最小的 seam 权值。在计算的过程中记录 seam 的路径。

3 使用不同算子进行测试

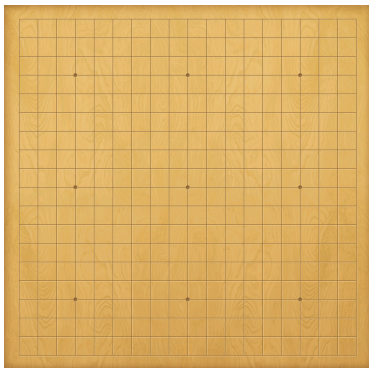
笔者使用了 Sobel 算子和拉普拉斯算子。以下是效果：

原图 :(之后横纵都 cut 1%)

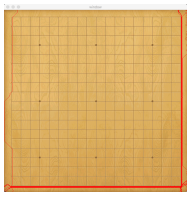
Laplacian :

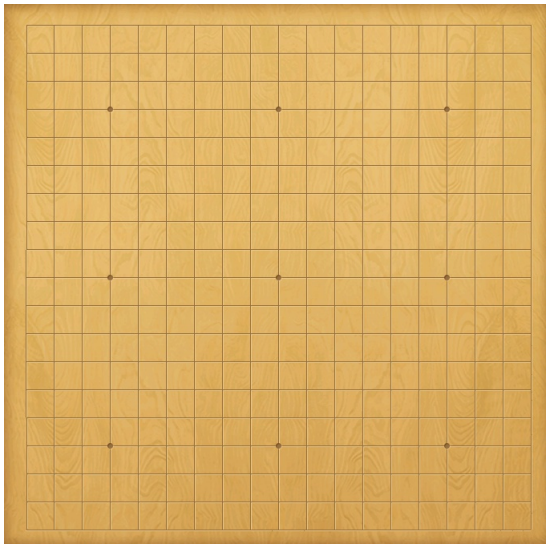
Sobel :

原图 :(之后横纵都 cut 1%)

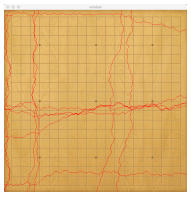


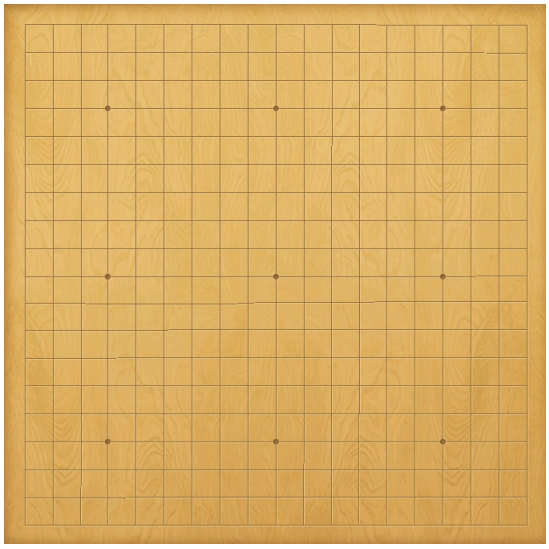
Laplacian :





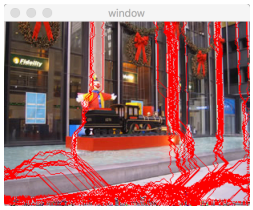
Sobel：



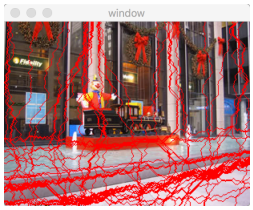


原图：(之后横纵都 cut 10%)

Laplacian：

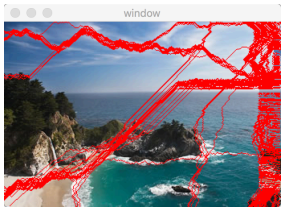


Sobel :

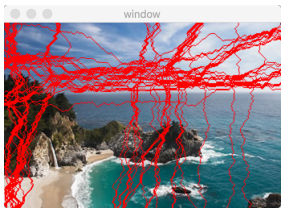


原图 :(之后横纵都 cut 10%)

Laplacian :



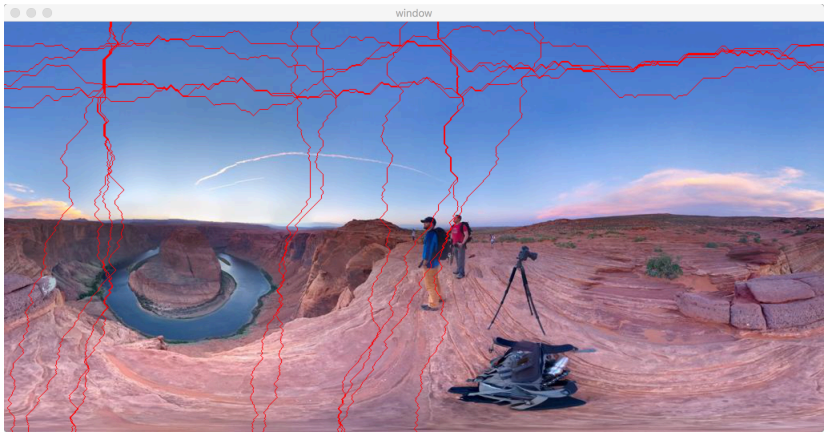
Sobel :



原图 :(之后横纵都 cut 1%)

Laplacian :

Sobel :



原图 :(之后横纵都 cut 0.1%)

Laplacian :

Sobel：

对应的代码段：

```
void calculateEnergy(Mat& image,double** energy,int row,int col)
{
  Mat imageXY8UC = image.clone();
  Mat imageX=Mat::zeros(image.size(),CV_8UC3);
    Mat imageY=Mat::zeros(image.size(),CV_8UC3);
    Mat imageXY=Mat::zeros(image.size(),CV_8UC3);
    Mat imageX8UC;
    Mat imageY8UC;
//     for(int i=0;i<image.rows;i++)
//     {
//         for(int j=0;j<image.cols;j++)
//         {
//             //通过指针遍历图像上每一个像素
//             for (int k = 0;k < 3;++k)
//               {
//                 imageX.at<Vec3b>(i,j)[k] =jueduizhi(
```

```
17  //            (i > 0 && j < image.cols-1 ? image.at<Vec3b>(i-1,j+1)[k] : 0)
18  //           + (j < image.cols-1 ? image.at<Vec3b>(i,j+1)[k]*2 : 0)
19  //           + (i < image.rows-1 && j < image.cols-1 ? image.at<Vec3b>(i+1,j+1)[k]
        : 0)
20  //           - (i > 0 && j > 0 ? image.at<Vec3b>(i-1,j-1)[k] : 0)
21  //           - (j > 0 ? image.at<Vec3b>(i,j-1)[k]*2 : 0)
22  //           - (i < image.rows-1 && j > 0 ? image.at<Vec3b>(i+1,j-1)[k] : 0)
23  //         );
24  //                imageY.at<Vec3b>(i,j)[k]=jueduizhi(
25  //           (i < image.rows-1 && j > 0 ? image.at<Vec3b>(i+1,j-1)[k] : 0)
26  //           + (i < image.rows-1 ? image.at<Vec3b>(i+1,j)[k]*2 : 0)
27  //           + (i < image.rows-1 && j < image.cols-1 ? image.at<Vec3b>(i+1,j+1)[k]
        : 0)
28  //           - (i > 0 && j > 0 ? image.at<Vec3b>(i-1,j-1)[k] : 0)
29  //           - (i > 0 ? image.at<Vec3b>(i-1,j)[k]*2 : 0)
30  //           - (i > 0 && j < image.cols-1 ? image.at<Vec3b>(i-1,j+1)[k] : 0)
31  //         );
32  //           }
33  //         }
34  //     }
35  //     addWeighted(imageX,0.5,imageY,0.5,0,imageXY);//融合X、Y方向
36  //     convertScaleAbs(imageX,imageX8UC);
37  //     convertScaleAbs(imageY,imageY8UC);
38  //     convertScaleAbs(imageXY,imageXY8UC);    //转换为8bit图像
39          for(int i=0;i<image.rows;i++)
40        {
41            for(int j=0;j<image.cols;j++)
42            {
43                //通过指针遍历图像上每一个像素
44                for (int k = 0;k < 3;++k)
45                  {
46                    imageXY8UC.at<Vec3b>(i,j)[k] =jueduizhi(
47                - (i > 0 ? image.at<Vec3b>(i-1,j)[k] : 0)
48                - (i < image.rows-1 ? image.at<Vec3b>(i+1,j)[k]*2 : 0)
49                - (j < image.cols-1 ? image.at<Vec3b>(i,j+1)[k] : 0)
50                - (j > 0 ? image.at<Vec3b>(i,j-1)[k] : 0)
51                + (image.at<Vec3b>(i,j)[k]*4)
52              );
53                }
54            }
55          }
56    for (int i = 0;i < row;++i)
57    {
58      for (int j = 0;j < col;++j)
59      {
60        energy[i][j] = 0;
61        for (int k = 0;k < 3;++k)
62        {
63          energy[i][j] += (int)(imageXY8UC.at<cv::Vec3b>(i,j)[k]);
64        }
65        // energy[i][j] += (int)(imageXY8UC.at<uchar>(i,j));
66      }
67    }
68  }
```

4 图像放大、双向缩放

双向缩放包括了图像放大，所以这里只写双向缩放了。

先说双向缩放的缩小。不同于单向缩放，双向缩放要考虑两边的放大和缩小，所以不能直接用直接暴力的 dp 来实现。论文中给出的算法是对于每次 cut 掉 r 行 c 列的图来 dp，新的图可以是被 cut 掉

r-1 行 c 列后又 cut 一行的结果，也可以是被 cut 掉 r 行 c-1 列又 cut 一列的结果。比较这两者之前的 cost 加上 seam 掉那一行或者一列的结果，我们就得到了新的 cost，使用这个 cost 我们选择出了新的图，并计算出行列 carve 之后的结果，用这样的结果再去做新的 dp。下面是代码：

```
void getInfo(T& picture,bool** choosed = NULL)
{
  int row = picture.pic.rows;
  int col = picture.pic.cols;
  double** energy = new double*[row];
  for (int i = 0;i < row;++i)
    energy[i] = new double[col];
  calculateEnergy(picture.pic,energy,row,col);
  if (choosed != NULL)
    for (int i = 0;i < row;++i)
      for (int j = 0;j < col;++j)
        if (choosed[i][j]) energy[i][j] = -INT_MAX;
  Node** seam = new Node*[row];
  for (int i = 0;i < row;++i)
    seam[i] = new Node[col];
  for (int i = 0;i < row;++i)
    for (int j = 0;j < col;++j)
      seam[i][j].number = INT_MAX;
  DP(seam,energy,row,col);
  int removenewPoint = calculateMin(seam,row,col,picture.cseam);
  type nowType = Col;
  picture.colpic = removeLine(seam,picture.pic,removenewPoint,row,col,picture.
      colseam,nowType);
  picture.pic = picture.pic.t(),nowType = Row;
  row = picture.pic.rows;
  col = picture.pic.cols;
  double** energyr = new double*[row];
  for (int i = 0;i < row;++i)
    energyr[i] = new double[col];
  calculateEnergy(picture.pic,energyr,row,col);
  if (choosed != NULL)
    for (int i = 0;i < col;++i)
      for (int j = 0;j < row;++j)
        if (choosed[i][j]) energyr[j][i] = -INT_MAX;
  Node** seamr = new Node*[row];
  for (int i = 0;i < row;++i)
    seamr[i] = new Node[col];
  for (int i = 0;i < row;++i)
    for (int j = 0;j < col;++j)
      seamr[i][j].number = INT_MAX;
  DP(seamr,energyr,row,col);
  removenewPoint = calculateMin(seamr,row,col,picture.rseam);
  picture.rowpic = removeLine(seamr,picture.pic,removenewPoint,row,col,picture.
      rowseam,nowType).t();
  picture.pic = picture.pic.t();
}
vector<vector<T> > dpphoto;
    vector<T> gg(totcol + 1);
    for (int i = 0;i < totrow + 1;++i)
      dpphoto.push_back(gg);
    dpphoto[0][0].pic = input;
    dpphoto[0][0].nowgain = 0;
    getInfo(dpphoto[0][0]);
    for (int i = 0;i <= totrow;++i)
      for (int j = 0;j <= totcol;++j)
      {
        if (i == 0 && j == 0) continue;
        else if (i == 0)
        {
          dpphoto[i][j].pic = dpphoto[i][j-1].colpic;
```

```
59            dpphoto[i][j].nowgain += dpphoto[i][j-1].cseam;
60            dpphoto[i][j].choose = Col;
61            getInfo(dpphoto[i][j]);
62          }
63          else if (j == 0)
64          {
65            dpphoto[i][j].pic = dpphoto[i-1][j].rowpic;
66            dpphoto[i][j].nowgain += dpphoto[i-1][j].rseam;
67            dpphoto[i][j].choose = Row;
68            getInfo(dpphoto[i][j]);
69          }
70          else
71          {
72            if (dpphoto[i-1][j].nowgain + dpphoto[i-1][j].rseam > dpphoto[i][j-1].
                  nowgain + dpphoto[i][j-1].cseam)
73            {
74              dpphoto[i][j].nowgain = dpphoto[i][j-1].nowgain + dpphoto[i][j-1].cseam
                    ;
75              dpphoto[i][j].choose = Col;
76              dpphoto[i][j].pic = dpphoto[i][j-1].colpic;
77              getInfo(dpphoto[i][j]);
78            }
79            else
80            {
81              dpphoto[i][j].nowgain = dpphoto[i-1][j].nowgain + dpphoto[i-1][j].rseam
                    ;
82              dpphoto[i][j].choose = Row;
83              dpphoto[i][j].pic = dpphoto[i-1][j].rowpic;
84              getInfo(dpphoto[i][j]);
85            }
86          }
87        }
88      Mat seamPic = dpphoto[totrow][totcol].pic;
89      imwrite("result.jpg",dpphoto[totrow][totcol].pic);
90      int r = totrow,c = totcol;
91      while (r >= 0 && c >= 0)
92      {
93        if (r == 0 && c == 0) break;
94        // printf("%d %d\n",r,c);
95        if (dpphoto[r][c].choose == Col)
96        {
97          --c;
98          if (c < 0) break;
99          Mat pic = Mat(seamPic.rows,seamPic.cols+1,CV_8UC3);
100         for (int j = seamPic.rows-1;j >= 0;--j)
101         {
102           int num = 0;
103           for (int p = 0;p < seamPic.cols;++p)
104             if (p == dpphoto[r][c].colseam[j].x)
105             {
106               pic.at<Vec3b>(dpphoto[r][c].colseam[j].y,num++) = Vec3b(0,0,255);
107               pic.at<Vec3b>(dpphoto[r][c].colseam[j].y,num++) = seamPic.at<Vec3b>(
                      dpphoto[r][c].colseam[j].y,p);
108             }
109             else
110               pic.at<Vec3b>(dpphoto[r][c].colseam[j].y,num++) = seamPic.at<Vec3b>(
                      dpphoto[r][c].colseam[j].y,p);
111         }
112         seamPic = pic;
113       }
114       else
115       {
116         --r;
```

```
        if (r < 0) break;
        Mat pic = Mat(seamPic.rows+1,seamPic.cols,CV_8UC3);
        for (int j = seamPic.cols-1;j >= 0;--j)
        {
          int num = 0;
          for (int p = 0;p < seamPic.rows;++p)
            if (p == dpphoto[r][c].rowseam[j].y)
            {
              pic.at<Vec3b>(num++,dpphoto[r][c].rowseam[j].x) = Vec3b(0,0,255);
              pic.at<Vec3b>(num++,dpphoto[r][c].rowseam[j].x) = seamPic.at<Vec3b>(p
                  ,dpphoto[r][c].rowseam[j].x);
            }
            else
              pic.at<Vec3b>(num++,dpphoto[r][c].rowseam[j].x) = seamPic.at<Vec3b>(p
                  ,dpphoto[r][c].rowseam[j].x);
        }
        seamPic = pic;
      }
    }
    imshow("window",seamPic);
    waitKey(0);
```

getInfo 通过当前图得到 seam 掉一行或一列后的图片，seam 掉一行的路径，seam 掉一列的路径，seam 掉一行的 cost，seam 掉一列的 cost，上一次 choose 了 seam 掉一行还是一列，当前的图片的 seam 权值。

然后主函数里面我们就对图片进行 dp，计算出切割掉 totrow,totcol 行之后的图片，然后从这张图片再还原 seam。

效果如算子测试中图所示。

双向放大的方法是，对行列分别放大。因为行列等价，我们这里只说列的放大方法。对于要放大的列数了 k，先找出这张图的 k 条 seam，然后把这 k 条 seam duplicate。对于找 k 条 seam，我的方法是每次把找到的 seam 的 energy 函数调得很大，这样就可以直接找到别的 seam 而不找这一条了。这样我们就能找到 k 条 seam 了。

代码如下：

```
1   int totrow= atof(argv[4])*temp.rows;
2   int totcol = atof(argv[3])*temp.cols;
3   double** energy = new double*[row];
4   for (int i = 0;i < row;++i)
5     energy[i] = new double[col];
6   unsigned short** addtime = new unsigned short*[row];
7   for (int i = 0;i < row;++i)
8     addtime[i] = new unsigned short[col];
9   for (int i = 0;i < row;++i)
10    for (int j = 0;j < col;++j)
11      addtime[i][j] = 0;
12  calculateEnergy(input,energy,row,col);
13  Node** seam = new Node*[row];
14  for (int i = 0;i < row;++i)
15    seam[i] = new Node[col];
16  double meiyongde = 0;
17  while (totcol--)
18  {
19    for (int i = 0;i < row;++i)
20      for (int j = 0;j < col;++j)
21        seam[i][j].number = INT_MAX;
22    DP(seam,energy,row,col);
23    int removenewPoint = calculateMin(seam,row,col,meiyongde);
24    int t = row-1;
25    while (removenewPoint != -1)
26    {
27      energy[t][removenewPoint] = INT_MAX;
28      addtime[t][removenewPoint]++;
```

```
29          removenewPoint = seam[t][removenewPoint].last;
30          t--;
31        }
32      Mat amplifyPic = Mat(temp.rows,temp.cols + atof(argv[3])*temp.cols,CV_8UC3);
33      for (int i = 0;i < temp.rows;++i)
34        for (int j = 0,k = 0;j < temp.cols;++j)
35          for (int p = 0;p <= addtime[i][j];++p)
36            amplifyPic.at<Vec3b>(i,k++) = temp.at<Vec3b>(i,j);
37      amplifyPic = amplifyPic.t();
38      row = amplifyPic.rows;
39      col = amplifyPic.cols;
40      double** energyr = new double*[row];
41      for (int i = 0;i < row;++i)
42        energyr[i] = new double[col];
43      calculateEnergy(amplifyPic,energyr,row,col);
44      Node** seamr = new Node*[row];
45      for (int i = 0;i < row;++i)
46        seamr[i] = new Node[col];
47      unsigned short** addtimenew = new unsigned short*[row];
48      for (int i = 0;i < row;++i)
49        addtimenew[i] = new unsigned short[col];
50      for (int i = 0;i < row;++i)
51        for (int j = 0;j < col;++j)
52          addtimenew[i][j] = 0;
53      while (totrow--)
54      {
55        for (int i = 0;i < row;++i)
56          for (int j = 0;j < col;++j)
57            seamr[i][j].number = INT_MAX;
58        DP(seamr,energyr,row,col);
59        int removenewPoint = calculateMin(seamr,row,col,meiyongde);
60        int t = row-1;
61        while (removenewPoint != -1)
62        {
63          energyr[t][removenewPoint] = INT_MAX;
64          addtimenew[t][removenewPoint]++;
65          removenewPoint = seamr[t][removenewPoint].last;
66          t--;
67        }
68      }
69      Mat amplify = Mat(temp.cols + atof(argv[3])*temp.cols,temp.rows + atof(argv[4])
            *temp.rows,CV_8UC3);
70      for (int i = 0;i < amplifyPic.rows;++i)
71        for (int j = 0,k = 0;j < amplifyPic.cols;++j)
72          if (addtimenew[i][j] == 0)
73            amplify.at<Vec3b>(i,k++) = amplifyPic.at<Vec3b>(i,j);
74          else
75          {
76            for (int p = 0;p <= addtimenew[i][j];++p)
77              amplify.at<Vec3b>(i,k++) = amplifyPic.at<Vec3b>(i,j);
78          }
79      amplify = amplify.t();
80      imshow("window",amplify);
81      imwrite("result.jpg",amplify);
82      waitKey(0);
```

5 对象移除

首先我们调用 opencv 自带的函数来选出要移除的选取，记录在 maskremove 里面。然后将图中这些要 remove 的东西的 energy function 中的权值置为一个很大的负权值，这样就可以在寻找 seam 的时候每次都删除掉 remove 区域里的点了。至于选择删除行还是列，这是使用的是贪心，如果行的 seam 值更小，就删除行。对于行列都计算一遍即可。

```
// 2. 创建一个可交互的窗口
    Image showImg = input.clone(); // 拷贝一张图用于显示（因为需要在显示的图上面高亮标注，从而造成修改）
    cv::namedWindow("Draw ROI", CV_WINDOW_AUTOSIZE); // 新建一个窗口
    vector<vector<int > > maskRemove(row, vector<int >(col, 0)); // 希望获取的待删除选区
    MouseArgs *args = new MouseArgs(showImg, maskRemove, Color(0, 0, 255)); // 攒一个MouseArgs结构体用于交互
    cv::setMouseCallback("Draw ROI", onMouse, (void*)args); // 给窗口设置回调函数

    // 拖动鼠标作画
    while (1)
    {
      cv::imshow("Draw ROI", args->img);
      // 按 esc 键退出绘图模式，获得选区
      if (cv::waitKey(100) == 27)
        break;
    }
    // maskRemove[200][400] = 1;
    bool** choosed = new bool*[row];
    for (int i = 0;i < row;++i)
      choosed[i] = new bool[col];
    for (int i = 0;i < row;++i)
      for (int j = 0;j < col;++j)
          choosed[i][j] = (maskRemove[i][j] == 1);
    while(1)
    {
      row = temp.rows;
      col = temp.cols;
      bool finished = true;
      for (int i = 0;i < row;++i)
        for (int j = 0;j < col;++j)
        {
          // printf("233%d %d\n",i,j);
          if (choosed[i][j]) finished = false;
        }
      if (finished) break;
      T nowpic;
      nowpic.pic = temp;
      getInfo(nowpic,choosed);
      if (nowpic.rseam < nowpic.cseam)
      {
        temp = nowpic.rowpic;
        row = temp.rows;
        col = temp.cols;
        // printf("%d %d\n",row,col);
        bool** gg = new bool*[row];
        for (int i = 0;i < row;++i)
          gg[i] = new bool[col];
        for (int i = col-1;i;--i)
          for (int j = 0,k = 0;j < row;++j)
            if (nowpic.rowseam[col-1-i].y != j)
              gg[k++][i] = choosed[j][i];
        for (int i = 0;i < row + 1;++i)
          delete[] choosed[i];
        delete[] choosed;
```

```
54            choosed = gg;
55          }
56          else
57          {
58            temp = nowpic.colpic;
59            row = temp.rows;
60            col = temp.cols;
61            // printf("%d %d\n",row,col);
62            bool** gg = new bool*[row];
63            for (int i = 0;i < row;++i)
64              gg[i] = new bool[col];
65            for (int i = row-1;i;--i)
66              for (int j = 0,k = 0;j < col;++j)
67                if (nowpic.colseam[row-1-i].x != j)
68                  gg[i][k++] = choosed[i][j];
69            for (int i = 0;i < row;++i)
70              delete[] choosed[i];
71            delete[] choosed;
72            choosed = gg;
73          }
74        }
75        // 4. 垃圾回收
76        cv::setMouseCallback("Draw ROI", NULL, NULL); // 取消回调函数
77        delete args; // 垃圾回收
78        // waitKey(0);
79
80        // imshow("window",temp);
81        imwrite("result.jpg",temp);
82        waitKey(0);
```

代码流程也就像我说的一样，先获取 remove 区域，然后对于 remove 区域中的点的 energy 设一个负权值，然后不停地删除 seam 直到删完 remove 区域中的点。

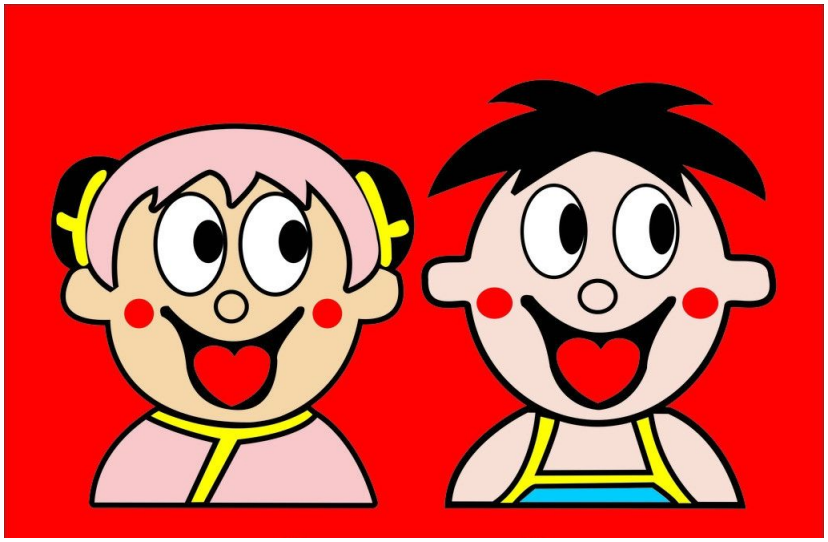
下面是效果：

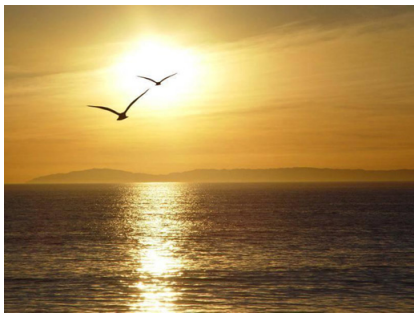
6 自选例子

以下均使用 Sobel 算子。

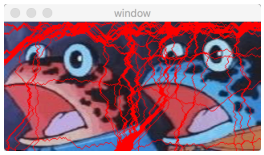
双向移除：

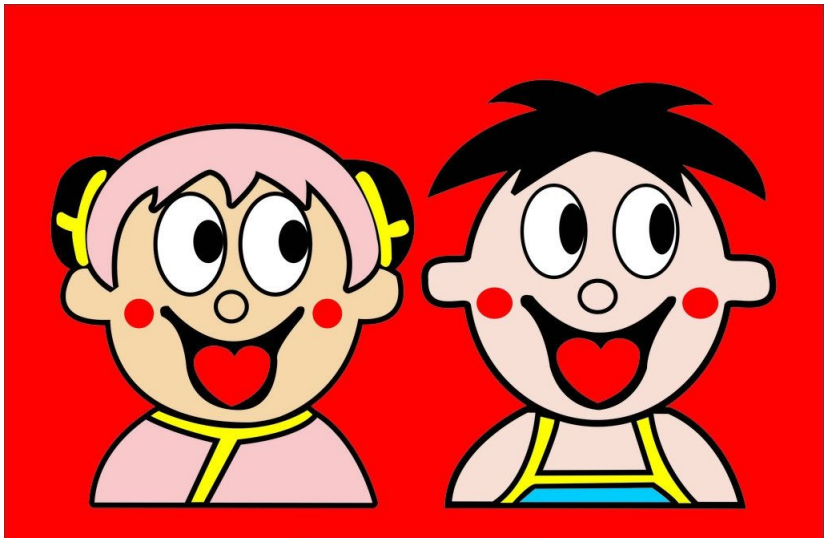
原图：

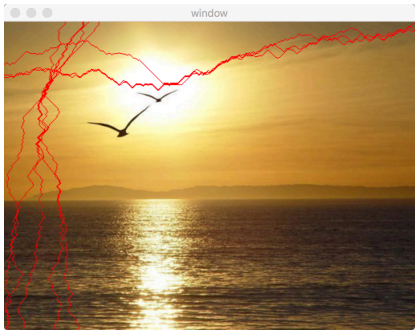


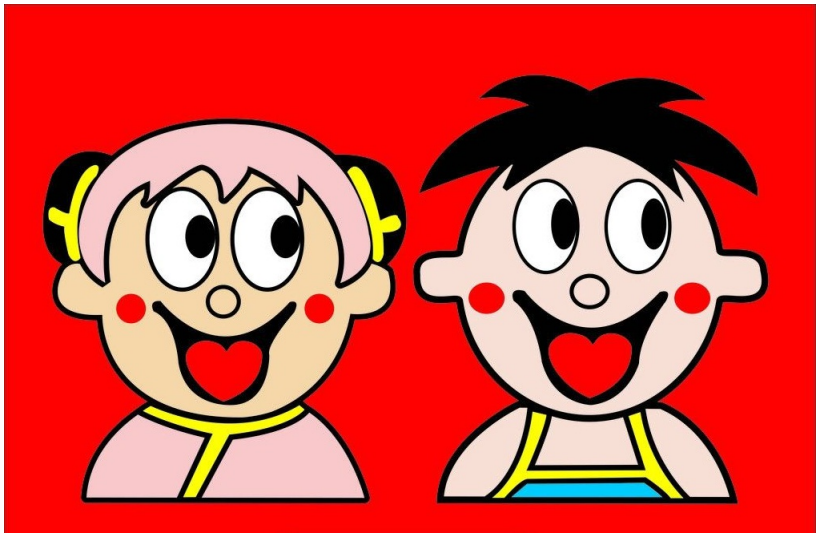


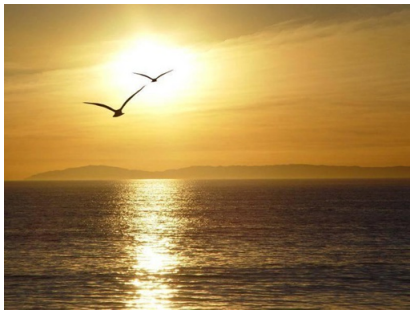
双向移除后 10%：






window





放大：